DevOps Bằng Hình

Học DevOps qua sơ đồ trực quan

Do Cao Hieu

Table of Contents

V — BẢO MẬT · GOVERNANCE · THỰC HÀNH

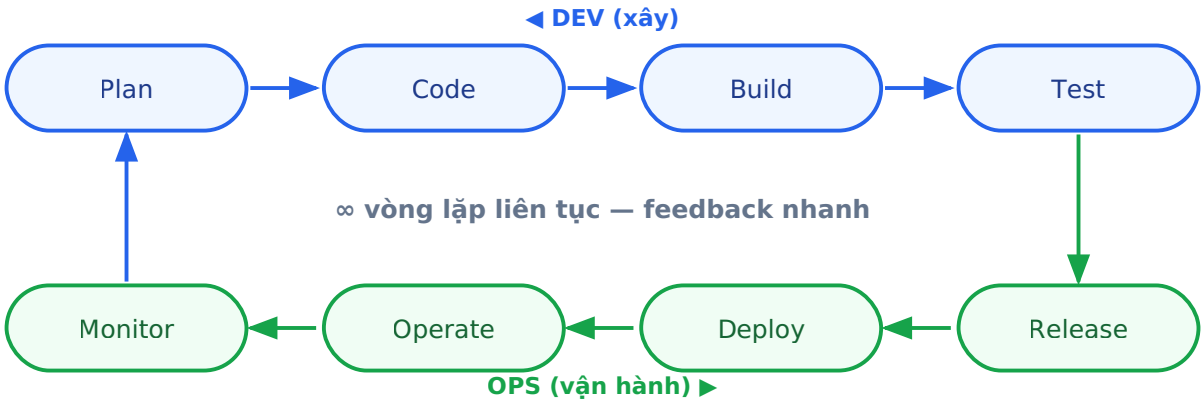
PHẦN

I — Nền tảng & Văn hóa

01

Văn hóa DevOps

DevOps không phải 1 tool — là văn hóa + thực hành kết hợp Dev (phát triển) và Ops (vận hành) để ship nhanh mà vẫn ổn định. Vòng đời chạy liên tục, feedback nhanh.



Thuật ngữ

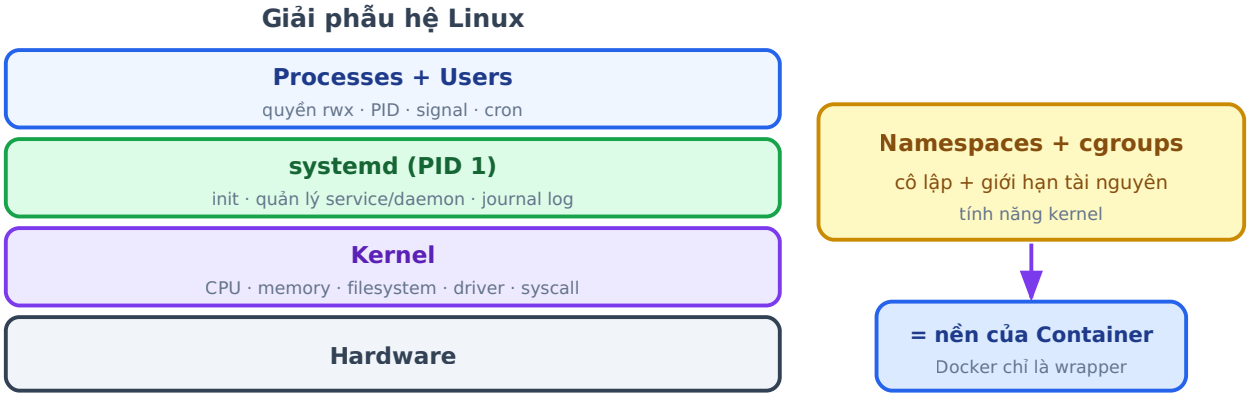
Thuật ngữ	Nghĩa
DevOps	Văn hóa + thực hành nối Dev và Ops; tự động hoá để giao hàng nhanh, an toàn
CALMS	5 trụ: **C**ulture · **A**utomation · **L**ean · **M**easurement · **S**haring
Shift Left	Đẩy test/security về sớm (lúc code) thay vì cuối
Blameless culture	Sự cố → mổ xẻ quy trình, không đổ lỗi cá nhân
You build it, you run it	Team viết code tự chịu trách nhiệm vận hành
Lean	Cắt lãng phí, giao giá trị nhỏ + liên tục
Feedback loop	Vòng phản hồi nhanh giữa các giai đoạn
Continuous everything	CI / CD / Continuous monitoring — mọi thứ liên tục

> DevOps = **người + quy trình + công cụ**. Tool chỉ là phần ngọn; văn hoá chia sẻ và đo lường mới là gốc.

02

Linux & OS Fundamentals

Nền móng của DevOps. Hầu hết server chạy Linux; container cũng dựa trên tính năng kernel Linux (namespaces + cgroups). Phải thạo shell, process, permission.



Package manager (apt/yum) · shell (bash) · SSH · /etc config · systemctl/journalctl là công cụ hằng ngày.

Thuật ngữ

Thuật ngữ	Nghĩa
Kernel	Lõi OS: quản CPU, memory, filesystem, driver
systemd	Init (PID 1), quản lý service/daemon
Process / PID	Tiến trình đang chạy, có ID
Signal	Tín hiệu điều khiển process (SIGTERM, SIGKILL)
Daemon	Tiến trình nền (sshd, nginx)
File permission	rwx cho user/group/other (chmod, chown)
Package manager	apt · yum/dnf · apk cài phần mềm
Shell	Bash/Zsh — giao diện dòng lệnh
cron	Lập lịch chạy job định kỳ
Namespace	Cô lập view (pid, net, mount...) → nền container
cgroups	Giới hạn tài nguyên (CPU/mem)
Filesystem / mount	Cây thư mục, gắn ổ đĩa
journald / syslog	Log hệ thống
env / PATH	Biến môi trường

Lệnh hằng ngày: `ps` · `top` · `systemctl` · `journalctl` · `df/du` · `netstat/ss` · `grep/sed/awk`.

03

Programming & Scripting

DevOps tự động hoá bằng code. Bash cho việc nhỏ, Python cho automation lớn, Go cho tool cloud-native, YAML/JSON cho config khai báo.

Ngôn ngữ nào cho việc gì

Bash	Python	Go	YAML / JSON
glue · CLI · cron	automation · tooling	cloud-native tools	config khai báo
thao tác file/process	API · data · glue lớn	CLI · binary tĩnh	K8s · CI · IaC
script ngắn < 100 dòng	ngôn ngữ #1 DevOps	K8s/Docker viết bằng Go	+ HCL (Terraform)

Kỹ năng kèm: regex · Git · API (curl) · jq/yq · template (Jinja/Go tmpl)

Quy tắc: tự động hoá việc lặp lại (eliminate toil), script idempotent, có error handling + log.

Thuật ngữ

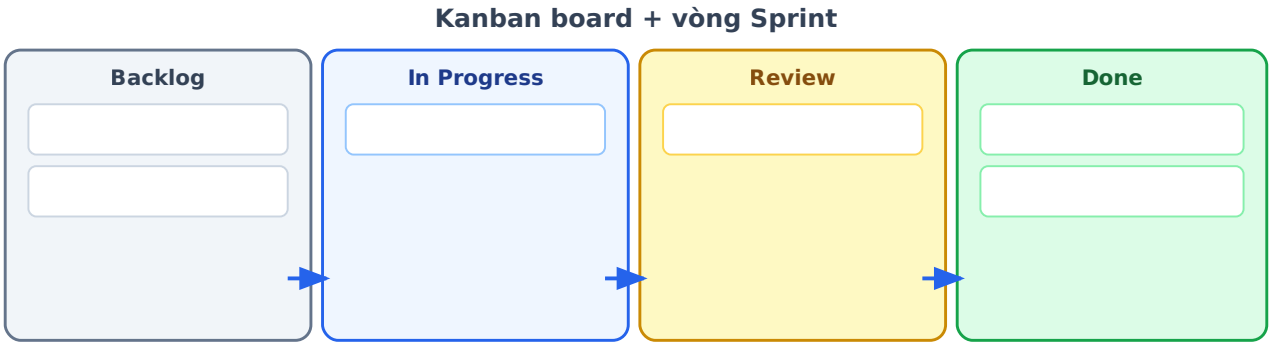
Thuật ngữ	Nghĩa
Bash	Shell scripting — glue, CLI, cron
Python	Ngôn ngữ automation #1 (tooling, API, data)
Go	Cho CLI/tool cloud-native (K8s, Docker viết bằng Go)
YAML / JSON	Định dạng config khai báo
HCL	Ngôn ngữ cấu hình Terraform
Regex	Khớp mẫu chuỗi
jq / yq	Truy vấn JSON / YAML từ CLI
Template	Sinh config động (Jinja2, Go template)
Idempotent script	Chạy lại an toàn, không tạo trùng
Exit code	Mã thoát 0 = ok, ≠0 = lỗi
stdin/stdout/stderr	Luồng vào/ra/lỗi để pipe
SDK / API client	Gọi dịch vụ cloud bằng code

> Quy tắc: tự động hoá việc lặp lại (eliminate toil), luôn có error handling + log.

04

Agile, Lean & Collaboration

DevOps là văn hoá → cách team tổ chức công việc quan trọng. Agile (Scrum/Kanban) giao giá trị nhỏ + liên tục; ChatOps đưa thao tác vào kênh chat.



Scrum: sprint 1-2 tuần · standup · planning · retro · velocity

Kanban: dòng chảy liên tục + giới hạn WIP. ChatOps: thao tác ngay trong Slack/Discord.

Thuật ngữ

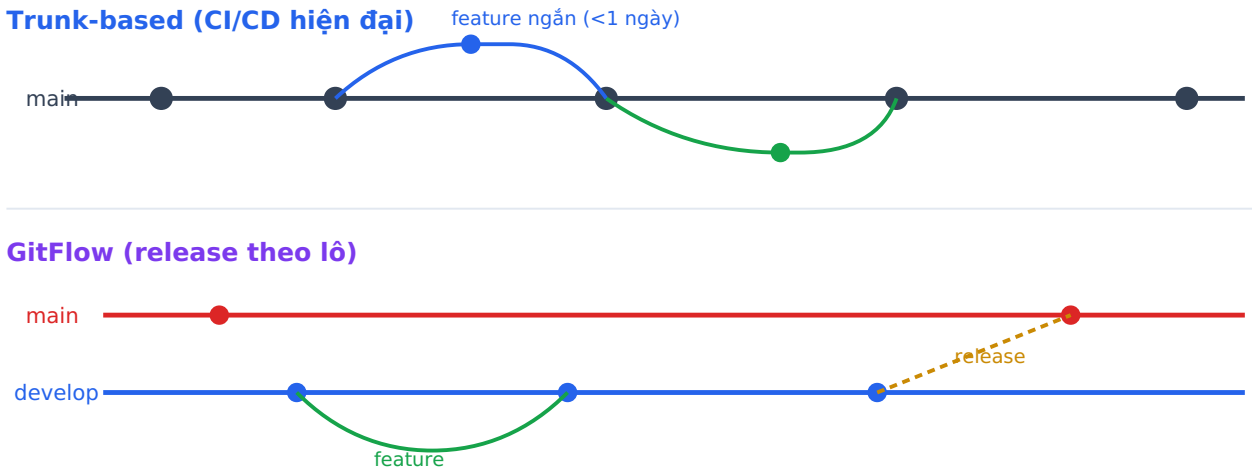
Thuật ngữ	Nghĩa
Agile	Giao giá trị nhỏ, lặp, thích ứng
Scrum	Khung làm việc theo sprint (1-2 tuần)
Kanban	Dòng chảy liên tục + giới hạn WIP
Sprint	Chu kỳ làm việc cố định
Backlog	Danh sách việc chờ làm
Standup	Họp nhanh hằng ngày
Retrospective	Họp rút kinh nghiệm cuối sprint
Velocity	Tốc độ hoàn thành của team
WIP limit	Giới hạn việc đang làm cùng lúc
ChatOps	Vận hành ngay trong Slack/Discord
Ticketing	Theo dõi task/issue (Jira, Linear)
ADR	Ghi lại quyết định kiến trúc

Tool: Jira · Linear · Trello · Slack/Discord · GitHub Projects.

05

Version Control & Git Workflows

Nền tảng của mọi quy trình DevOps. Cách team nhánh/merge code quyết định tốc độ giao hàng. Trunk-based hợp CI/CD liên tục; GitFlow hợp release theo lô.



Thuật ngữ

Thuật ngữ	Nghĩa
Repository	Kho chứa code + lịch sử
Commit	1 ảnh chụp thay đổi có message
Branch	Nhánh tách ra để làm tính năng
Merge vs Rebase	Gộp giữ lịch sử rẽ nhánh vs viết lại thẳng hàng
PR / MR	Pull/Merge Request — yêu cầu gộp + review
Trunk-based	Nhánh ngắn, merge vào main liên tục (hợp CI/CD)
GitFlow	main + develop + feature/release/hotfix (release lô)
GitHub Flow	main + feature branch + PR, deploy ngay
Monorepo vs Polyrepo	1 repo cho tất cả vs mỗi service 1 repo
SemVer	Đánh version MAJOR.MINOR.PATCH
Conventional Commits	Quy ước message: `feat:` `fix:` `chore:`
Tag	Đánh dấu 1 mốc (release)

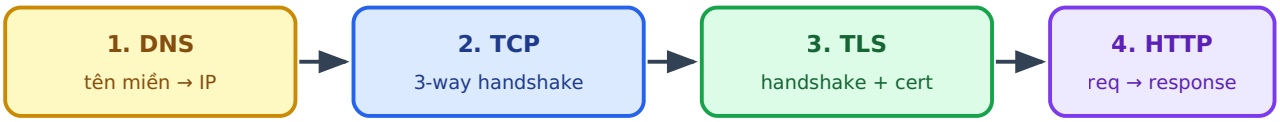
Tool: Git · GitHub · GitLab · Bitbucket.

06

Networking & Protocols

DevOps phải hiểu đường đi của 1 request ở tầng protocol để debug, đặt cache, bảo mật đúng chỗ. Một request HTTPS đi qua: DNS → TCP → TLS → HTTP.

Vòng đời 1 request HTTPS



Tầng (mô hình TCP/IP):

API Gateway / Reverse Proxy đứng trước app: định tuyến, rate-limit, auth, TLS termination.

Thuật ngữ

Thuật ngữ	Nghĩa
DNS	Phân giải tên miền → IP (A, CNAME, MX records)
TCP / UDP	Giao thức tin cậy (bắt tay) vs nhanh không đảm bảo
3-way handshake	SYN → SYN-ACK → ACK mở kết nối TCP
TLS / SSL	Mã hoá + xác thực bằng certificate
HTTP/HTTPS	Giao thức web; method GET/POST, status 2xx/4xx/5xx
Port	Cổng dịch vụ (80, 443, 22, 5432…)
SSH	Truy cập server an toàn qua khoá
Reverse Proxy	Đứng trước backend, định tuyến (Nginx)
API Gateway	Cổng API: auth, rate-limit, routing
Load Balancer	Phân tải L4 (TCP) / L7 (HTTP)
Firewall / NAT	Lọc traffic / dịch địa chỉ
OSI / TCP-IP model	Phân tầng mạng (App→Transport→Internet→Link)

Tool: dig · curl · nmap · Wireshark · Nginx · Envoy.

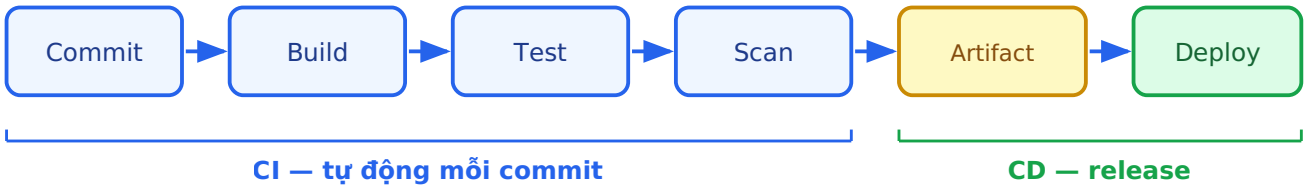
PHẦN

II — Build & Ship

07

CI / CD Pipeline

CI (Continuous Integration) — tự động build + test mỗi khi push code. **CD (Delivery)** — code luôn sẵn sàng, bấm nút mới lên prod. **CD (Deployment)** — tự động lên prod luôn, không cần bấm tay.



- Delivery = dừng trước Deploy, chờ người bấm duyệt.
- Deployment = mũi tên Deploy chạy tự động, không gián đoạn.

Thuật ngữ

Thuật ngữ	Nghĩa
CI (Continuous Integration)	Tự động merge + build + test mỗi khi push code
CD (Continuous Delivery)	Code luôn sẵn sàng deploy; bấm nút thủ công mới lên prod
CD (Continuous Deployment)	Như trên nhưng tự động lên prod, không cần bấm
Pipeline	Chuỗi stage tự động: build → test → scan → deploy
Stage / Job / Step	Các cấp đơn vị trong pipeline
Build	Biên dịch source thành thứ chạy được
Artifact	Sản phẩm build (image Docker, .jar) lưu ở registry
Runner / Agent	Máy thực thi các job của pipeline
Trigger	Điều kiện kích hoạt pipeline (push, PR, cron)
Cache	Lưu dependency để build nhanh hơn

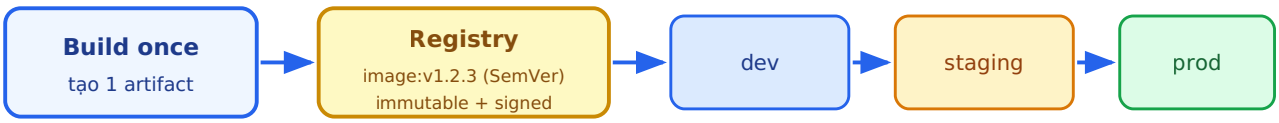
Tool: GitHub Actions · GitLab CI · Jenkins · CircleCI · Argo Workflows.

08

Artifacts, Registries & Supply Chain

Nguyên tắc vàng: **build once, promote everywhere** — build 1 artifact bất biến, đẩy cùng nó qua dev → staging → prod. Bảo vệ chuỗi cung ứng chống chèn mã độc.

Build once, promote everywhere — cùng 1 artifact qua mọi môi trường



Không build lại mỗi env → đảm bảo "cái test = cái chạy prod".
Supply chain: SBOM (kê khai thành phần) + ký số + provenance (SLSA) chống chèn mã độc.

Thuật ngữ

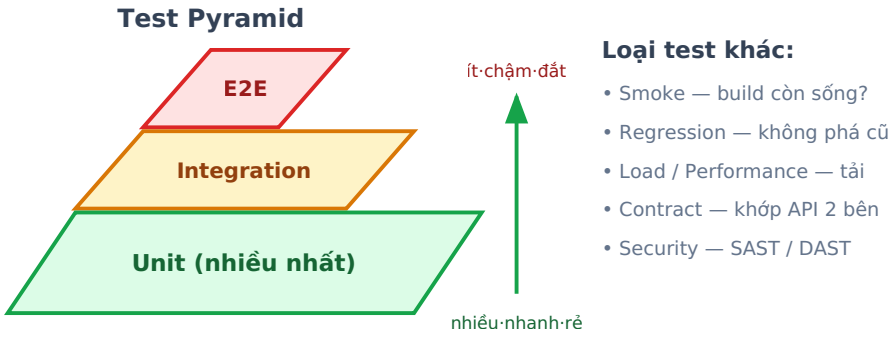
Thuật ngữ	Nghĩa
Artifact	Sản phẩm build (image, .jar, binary)
Registry	Kho lưu artifact (container/package)
Immutable artifact	Không sửa sau khi build; mỗi version là 1 bản
SemVer	MAJOR.MINOR.PATCH cho version
Promotion	Chuyển cùng artifact lên env cao hơn
Build once	Build 1 lần, dùng lại mọi env
SBOM	Software Bill of Materials — kê khai thành phần
Provenance / SLSA	Chứng minh nguồn gốc artifact
Signing	Ký số artifact chống giả mạo (cosign)
Supply chain security	Bảo vệ toàn chuỗi từ code → artifact → deploy
Pinning	Khoá version dependency cố định

Tool: Docker/OCI registry · GHCR/ECR · Artifactory · Nexus · Cosign · Syft/Grype.

09

Testing & Quality

Test tự động là cổng chất lượng trong CI/CD. Test pyramid: nhiều unit (nhanh, rẻ) ở đáy, ít e2e (chậm, đắt) ở đỉnh.



Shift Left: test sớm trong pipeline, fail fast.

Đáy rộng (unit) nhiều & nhanh; đỉnh hẹp (e2e) ít & chậm — cân bằng chi phí/độ tin cậy.

Thuật ngữ

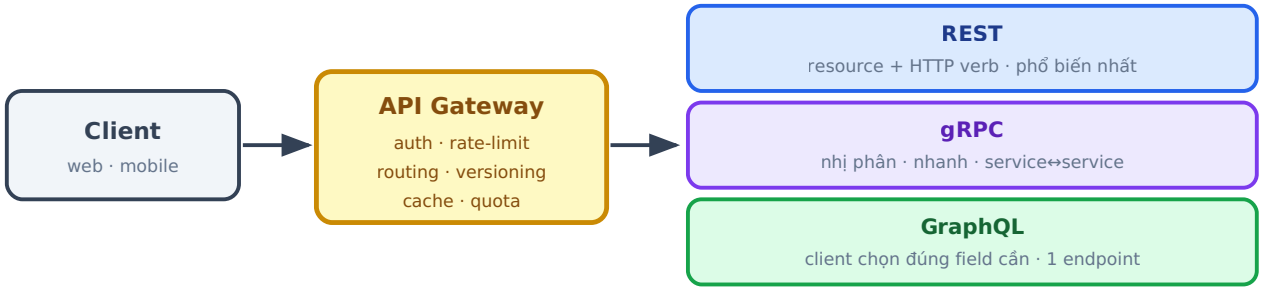
Thuật ngữ	Nghĩa
Unit test	Test 1 hàm/đơn vị nhỏ, isolated
Integration test	Test nhiều thành phần ghép lại
E2E test	Test luồng người dùng đầu-cuối
Test pyramid	Nhiều unit, ít e2e — cân chi phí/độ tin
Smoke test	Kiểm tra nhanh build có "sống" không
Regression test	Đảm bảo không phá tính năng cũ
Load / Performance test	Chịu tải, đo throughput/latency
Contract test	Khớp hợp đồng API giữa 2 service
Coverage	% code được test bao phủ
TDD / BDD	Viết test trước / theo hành vi
Flaky test	Test lúc pass lúc fail (không ổn định)
Mock / Stub	Giả lập dependency khi test

Tool: Jest/Vitest · Pytest · Playwright/Cypress · k6/JMeter · Pact.

10

API Design & Management

Service giao tiếp qua API. Chọn đúng style (REST/gRPC/GraphQL) và đặt API Gateway trước để lo auth, rate-limit, versioning.



Hợp đồng: OpenAPI/Swagger (REST) · Protobuf (gRPC) · Schema (GraphQL)

Versioning (v1/v2) · idempotency · pagination · status code chuẩn 2xx/4xx/5xx.

Thuật ngữ

Thuật ngữ	Nghĩa
REST	Resource + HTTP verb; phổ biến nhất
gRPC	Nhị phân, nhanh; hợp service↔service
GraphQL	Client chọn đúng field, 1 endpoint
API Gateway	Cổng: auth, rate-limit, routing, cache
OpenAPI / Swagger	Đặc tả hợp đồng REST
Protobuf	Định dạng schema cho gRPC
Versioning	v1/v2 để không phá client cũ
Rate limiting / Quota	Giới hạn số request
Idempotency	Gọi lại không gây tác dụng phụ kép
Pagination	Chia trang kết quả lớn
Webhook	Server gọi ngược client khi có sự kiện
Status code	2xx ok · 4xx lỗi client · 5xx lỗi server

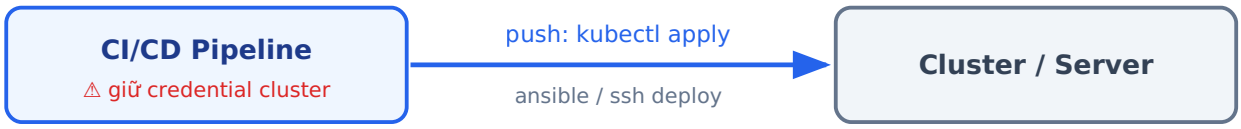
Tool: Kong · Apigee · AWS API Gateway · Postman · Swagger UI.

11

Push-based vs Pull-based

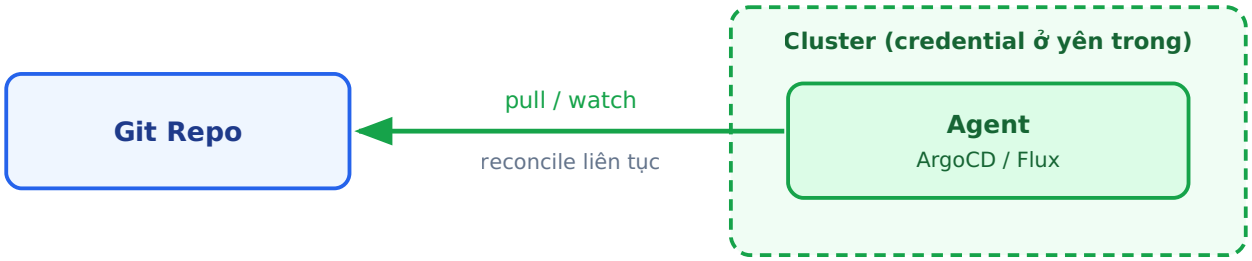
Hai mô hình đưa thay đổi vào môi trường target. Khác nhau ở **ai khởi xướng** và **credential nằm ở đâu**.

Push 📤 — CI chủ động đẩy



✅ Đơn giản, kiểm soát timing chính xác. ❌ CI giữ credential prod (rủi ro); không tự sửa khi state lệch (drift).

Pull 📥 — Agent tự kéo về



✅ Credential không rời cluster (an toàn); self-healing chống drift. ❌ Phức tạp hơn, có độ trễ poll.

> **1 câu:** Push = "tao đẩy cho mày". Pull = "tao tự lấy về". **GitOps thường chọn pull.**

Thuật ngữ

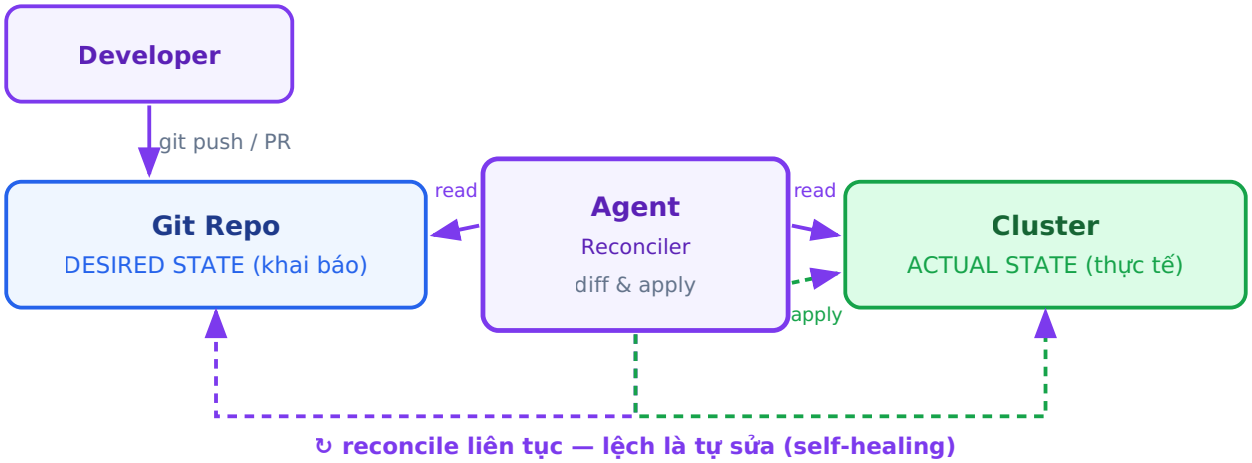
Thuật ngữ	Nghĩa
Push-based	CI server chủ động đẩy thay đổi vào target
Pull-based	Agent trong target tự kéo thay đổi từ Git về
Credential	Khoá/quyền truy cập môi trường target
Drift	Trạng thái thực tế lệch khỏi khai báo
Reconcile	Agent so sánh + đồng bộ actual về desired
Self-healing	Tự sửa khi bị sửa tay / lệch state
Poll vs Watch	Định kỳ hỏi vs lắng nghe thay đổi real-time

Tool (pull): ArgoCD · Flux. **Tool (push):** Jenkins · GitHub Actions + kubectl/ansible.

12

GitOps — Vòng lặp Reconcile

Git = single source of truth. Toàn bộ trạng thái hệ thống khai báo bằng file trong Git. Agent liên tục so sánh **desired state** (Git) vs **actual state** (Cluster), lệch là tự sửa.



4 nguyên tắc GitOps: (1) Declarative — mô tả "muốn gì". (2) Versioned — rollback = `git revert`. (3) Pulled — agent tự kéo. (4) Reconciled — sửa drift tự động.

Ai đó sửa tay trên cluster → agent phát hiện lệch desired state → kéo về đúng như Git. Đó là **self-healing**.

Thuật ngữ

Thuật ngữ	Nghĩa
GitOps	Git là single source of truth cho cả infra + app
Single source of truth	1 nguồn sự thật duy nhất (Git repo)
Desired state	Trạng thái mong muốn, khai báo trong Git
Actual state	Trạng thái thực tế trên cluster
Reconciliation	Vòng lặp so sánh + sửa cho khớp desired
Declarative	Mô tả "muốn gì", không phải "làm thế nào"
Rollback	Quay version = `git revert`
Drift detection	Phát hiện lệch và tự sửa

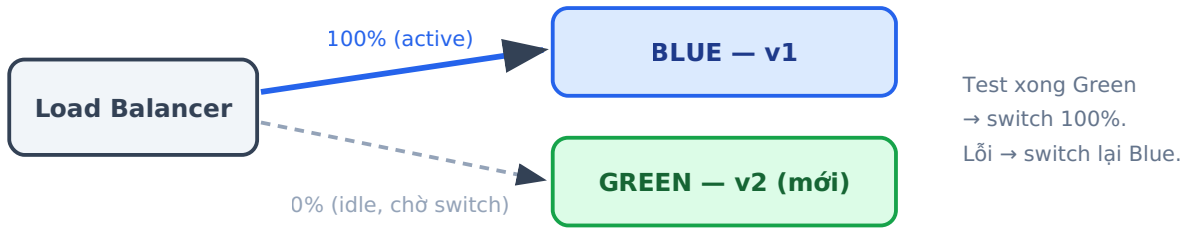
Tool: ArgoCD · Flux · Jenkins X.

13

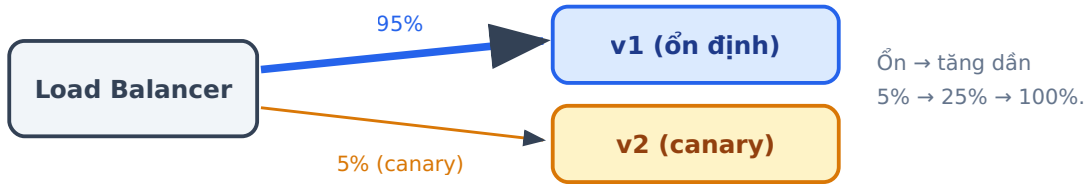
Deployment Strategies

Cách đưa version mới lên prod mà giảm thiểu downtime + rủi ro.

Blue-Green — đổi cờ tức thì

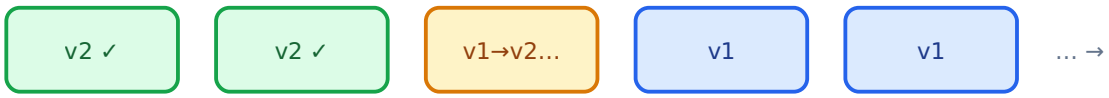


Canary — thả từ từ theo %



Rolling — thay thế lần lượt

Thay từng instance một, không downtime:



Thuật ngữ

Thuật ngữ	Nghĩa
Blue-Green	2 môi trường; switch traffic 100% qua bản mới, rollback = switch lại
Canary	Thả bản mới cho % nhỏ user, tăng dần nếu ổn
Rolling update	Thay thế dần từng instance, không downtime
A/B testing	Chia traffic theo tiêu chí để so sánh (mục đích kinh doanh)
Feature flag	Bật/tắt tính năng bằng config, không cần deploy lại
Rollback	Quay về version cũ khi lỗi

Thuật ngữ	Nghĩa
Zero-downtime deploy	Deploy không gián đoạn dịch vụ
Traffic shifting	Dịch chuyển % traffic giữa các version
Health check / Readiness	Kiểm tra instance sẵn sàng nhận traffic chưa

Tool: Argo Rollouts · Flagger · Spinnaker · service mesh (Istio).

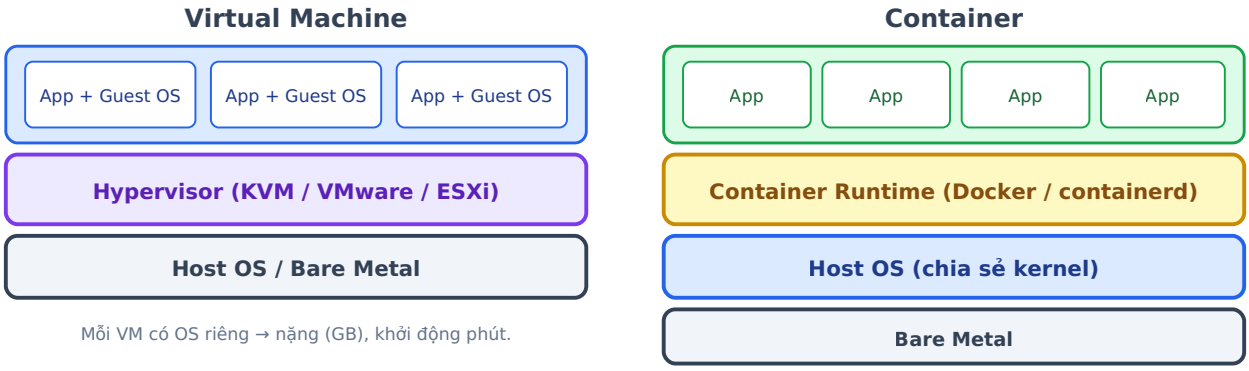
PHẦN

III — Hạ tầng & Compute

14

Virtualization & Compute

Tầng tính toán bên dưới container. VM ảo hoá cả phần cứng (có OS riêng, nặng); container chia sẻ kernel host (nhẹ). Bare metal chạy thẳng không ảo hoá.



Container chia sẻ kernel host → nhẹ (MB), khởi động giây. Bare metal = chạy thẳng không ảo hoá (hiệu năng max).

Thuật ngữ

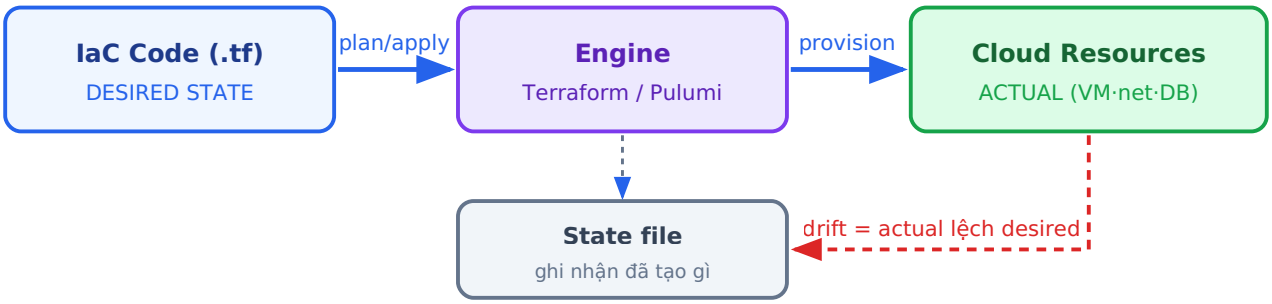
Thuật ngữ	Nghĩa
Bare Metal	Chạy thẳng trên phần cứng, không ảo hoá
Virtual Machine (VM)	Máy ảo có Guest OS riêng
Hypervisor	Lớp tạo/quản VM (KVM, VMware, Hyper-V)
Type 1 / Type 2	Hypervisor chạy thẳng HW / trên host OS
Guest OS / Host OS	OS trong VM / OS nền
Container	Chia sẻ kernel host, nhẹ (MB), nhanh
Image	Template tạo VM/container
Snapshot	Ảnh chụp trạng thái để khôi phục
Overcommit	Cấp vượt tài nguyên vật lý
Live migration	Chuyển VM đang chạy sang host khác
WASM	Sandbox nhẹ mới nổi, thay thế 1 phần container

Tool: KVM/QEMU · VMware · Proxmox · VirtualBox · Firecracker (microVM).

15

Infrastructure as Code (IaC)

Quản lý hạ tầng bằng **code/file** thay vì click tay. Khai báo trạng thái mong muốn (desired state), engine tự dựng. So sánh actual vs desired để phát hiện drift.



Idempotent: chạy lại nhiều lần → kết quả như nhau, không tạo trùng.

Thuật ngữ

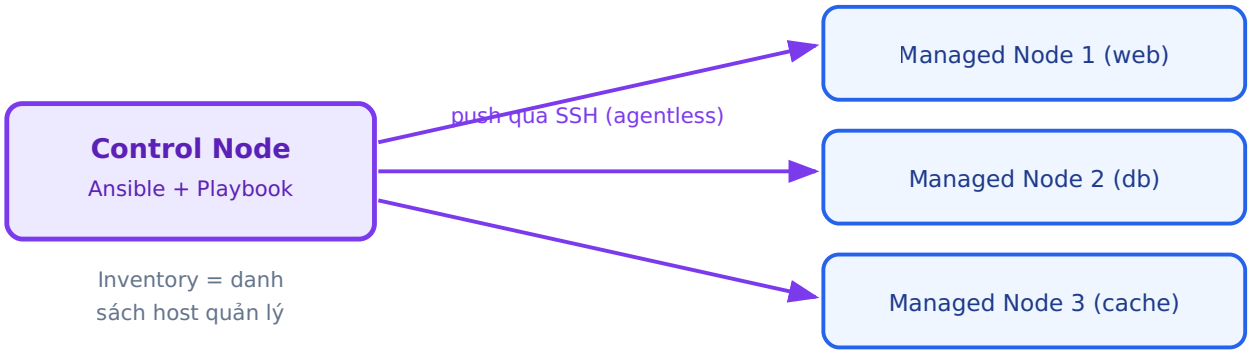
Thuật ngữ	Nghĩa
IaC	Hạ tầng định nghĩa bằng code, versioned trong Git
Declarative	Khai báo "muốn trạng thái gì" (Terraform, K8s YAML)
Imperative	Ra lệnh từng bước "làm thế nào" (script bash)
Idempotent	Chạy nhiều lần → kết quả như nhau, không tạo trùng
Provisioning	Cấp phát tài nguyên (VM, network, DB)
Drift	Thực tế lệch khỏi state khai báo (ai đó sửa tay)
State file	File lưu những gì IaC đang quản lý
Plan / Apply	Xem trước thay đổi (plan) rồi áp dụng (apply)
Module	Khối IaC tái sử dụng được
Immutable infra	Lỗi thì thay mới hoàn toàn, không sửa tại chỗ
Mutable infra	Sửa trực tiếp server đang chạy

Tool: Terraform · OpenTofu · Pulumi · CloudFormation · Crossplane.

16

Configuration Management

Tự động **cài đặt + cấu hình** phần mềm trên server (sau khi IaC đã dựng máy). Ansible chạy agentless qua SSH, đẩy playbook xuống nhiều host cùng lúc.



Thuật ngữ

Thuật ngữ	Nghĩa
Config management	Tự động cấu hình phần mềm/OS trên server
Playbook / Recipe / Manifest	File mô tả công việc (Ansible / Chef / Puppet)
Agentless	Không cần cài agent — Ansible dùng SSH
Agent-based	Phải cài daemon trên host (Puppet/Chef)
Inventory	Danh sách host cần quản lý
Role / Module	Khối tác vụ tái sử dụng
Idempotent	Áp lại cấu hình nhiều lần vẫn an toàn
Push vs Pull	Đẩy từ control node (Ansible) vs agent tự kéo (Puppet)

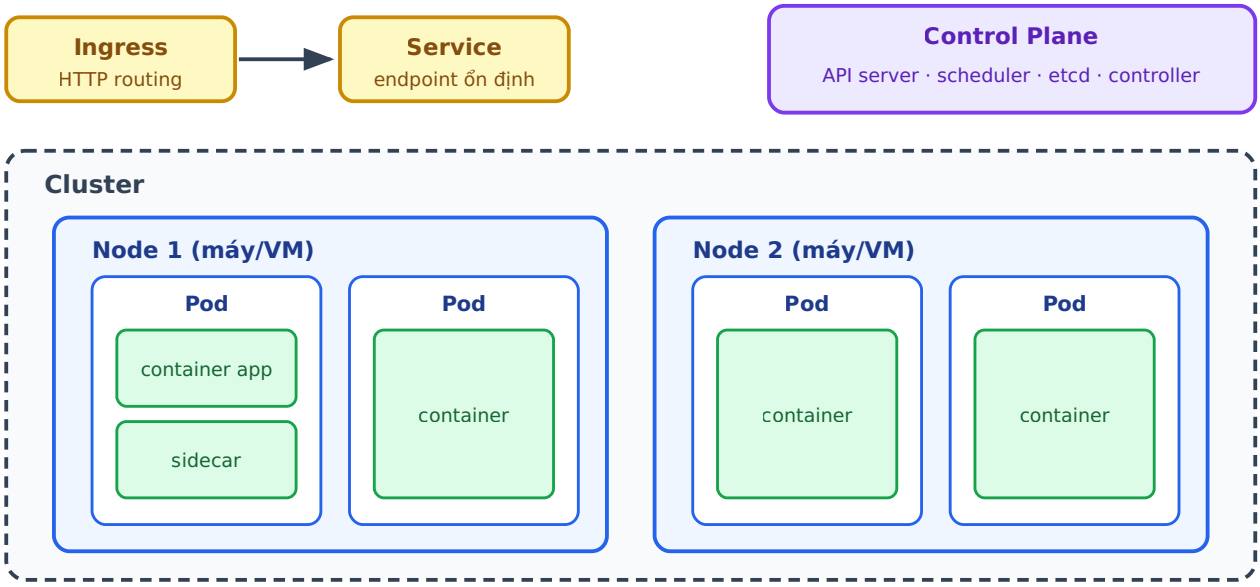
Tool: Ansible · Puppet · Chef · SaltStack.

> Khác IaC: IaC dựng *hạ tầng* (máy/network), Config Management cấu hình *bên trong* máy đó. Thường dùng chung.

17

Container & Orchestration

Container = đóng gói app + dependency, chạy isolated, nhẹ hơn VM. **Orchestration** (Kubernetes) = quản lý container ở quy mô lớn: scale, heal, schedule.



Thuật ngữ

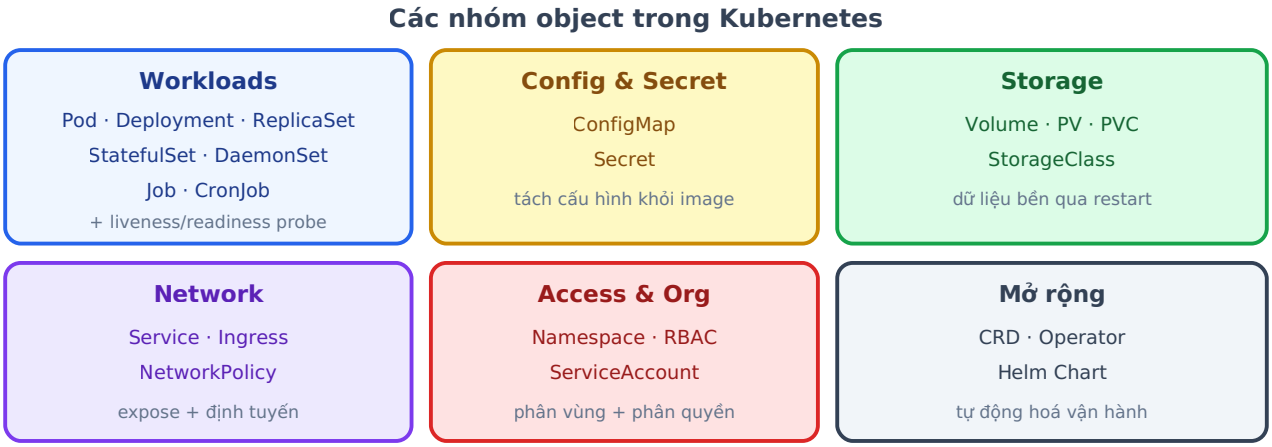
Thuật ngữ	Nghĩa
Container	Đóng gói app + dependency, isolated, nhẹ hơn VM
Image	Template bất biến để tạo container
Registry	Kho lưu image (Docker Hub, GHCR, ECR)
Orchestration	Quản lý container ở quy mô lớn (K8s)
Pod	Đơn vị nhỏ nhất (1+ container)
Node	Máy (VM/physical) chạy workload
Cluster	Tập hợp nodes
Deployment	Object quản lý replica + rolling update
Service	Endpoint mạng ổn định cho pods
Ingress	Định tuyến HTTP/HTTPS từ ngoài vào
Control Plane	API server · scheduler · etcd · controller
Helm / Chart	Package manager cho K8s
Sidecar	Container phụ chạy kèm (logging, proxy)
Service Mesh	Lớp quản lý giao tiếp service↔service (Istio)
HPA	Auto-scale theo CPU/memory

Tool: Docker · Kubernetes · containerd · Podman · Helm.

18

Kubernetes Deep-Dive

Ngoài Pod/Node/Service cơ bản, K8s có nhiều loại object chia theo nhóm chức năng. Hiểu nhóm nào làm gì giúp khai báo manifest đúng.



Thuật ngữ

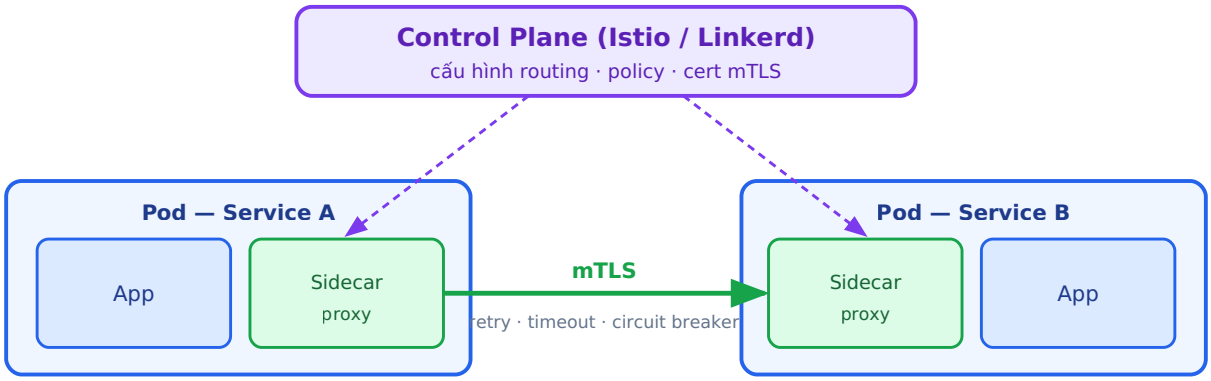
Object	Nhóm	Nghĩa
Deployment	Workload	Quản lý replica + rolling update (stateless)
StatefulSet	Workload	Cho app có state (DB), ID ổn định
DaemonSet	Workload	1 pod trên mỗi node (agent, log collector)
Job / CronJob	Workload	Chạy 1 lần / theo lịch
ConfigMap	Config	Cấu hình tách khỏi image
Secret	Config	Lưu dữ liệu nhạy cảm (base64)
PV / PVC	Storage	Volume bền + yêu cầu cấp phát
StorageClass	Storage	Lớp lưu trữ động
Service	Network	Endpoint ổn định cho pods
Ingress	Network	Định tuyến HTTP từ ngoài vào
NetworkPolicy	Network	Firewall giữa các pod
Namespace	Org	Phân vùng logic trong cluster
RBAC / ServiceAccount	Access	Phân quyền theo vai trò
Probe (liveness/readiness)	Workload	Kiểm tra pod sống / sẵn sàng
CRD / Operator	Mở rộng	Tự định nghĩa resource + tự động vận hành

Tool: kubectl · Helm · Kustomize · k9s · Lens.

19

Service Mesh

Khi có nhiều microservice, quản lý giao tiếp service↔service (mã hoá, retry, observability) trở nên phức tạp. Service mesh nhét 1 **sidecar proxy** cạnh mỗi service để gánh phần đó.



Sidecar gánh networking → app không phải viết logic retry/mã hoá/observability.

Thuật ngữ

Thuật ngữ	Nghĩa
Service Mesh	Lớp hạ tầng quản lý giao tiếp service↔service
Sidecar	Proxy chạy cạnh app trong cùng pod
Data plane	Tập các sidecar xử lý traffic thực
Control plane	Cấu hình + phân phối policy/cert cho sidecar
mTLS	Mã hoá + xác thực 2 chiều giữa service
Traffic splitting	Chia % traffic (canary, A/B)
Circuit breaker	Ngắt mạch khi service lỗi liên tục
Retry / Timeout	Tự thử lại / giới hạn chờ
Service discovery	Tự tìm địa chỉ service
Observability	Tự sinh metrics/traces cho mọi call

Tool: Istio · Linkerd · Consul · Cilium · Envoy (proxy).

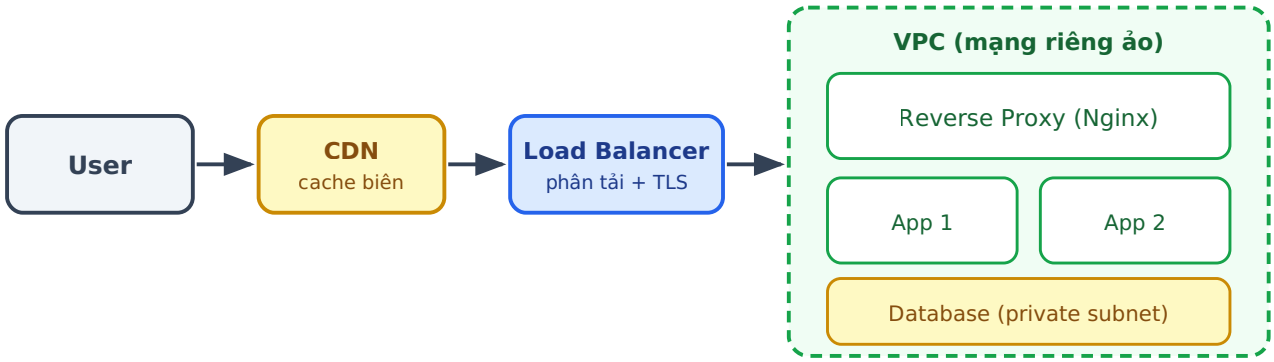
PHẦN

IV — Vận hành

20

Cloud & Networking

Đường đi 1 request: User → CDN → Load Balancer → Reverse Proxy → App (trong VPC) → DB. Hiểu các tầng mạng giúp đặt cache, bảo mật và scale đúng chỗ.



Thuật ngữ

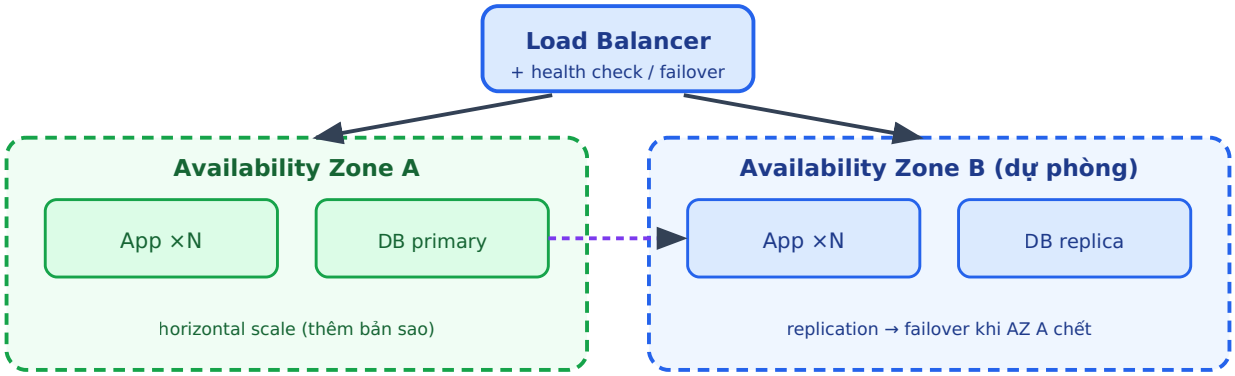
Thuật ngữ	Nghĩa
IaaS / PaaS / SaaS	3 mô hình dịch vụ cloud (hạ tầng / nền tảng / phần mềm)
Serverless / FaaS	Chạy code không quản server (Lambda, Cloud Run)
Multi-cloud / Hybrid	Nhiều cloud / kết hợp on-premise + cloud
Region / AZ	Vùng địa lý / Availability Zone (cách ly lỗi)
Load Balancer	Phân tải traffic ra nhiều backend
Reverse Proxy	Đứng trước backend, định tuyến (Nginx, Traefik)
CDN	Cache nội dung ở biên, gần user
VPC	Mạng riêng ảo trong cloud
Subnet (public/private)	Phân mảnh mạng; DB thường ở private
DNS	Phân giải tên miền → IP
Auto-scaling / Elasticity	Tự co giãn tài nguyên theo tải
Ingress / Egress	Traffic vào / ra

Tool: AWS · GCP · Azure · Cloudflare · Nginx · Traefik.

21

Scaling, HA & Disaster Recovery

Hệ thống phải co giãn theo tải và sống sót khi 1 phần chết. Trải workload qua nhiều AZ, có bản sao + failover, backup định kỳ.



RTO = bao lâu phục hồi xong · RPO = chấp nhận mất bao nhiêu dữ liệu

Redundancy + backup định kỳ + multi-region = nền của High Availability & Disaster Recovery.

Thuật ngữ

Thuật ngữ	Nghĩa
Horizontal scaling	Thêm nhiều instance (scale out)
Vertical scaling	Tăng CPU/RAM 1 instance (scale up)
Auto-scaling	Tự co giãn theo tải
Load balancing	Phân tải ra nhiều backend
High Availability (HA)	Không có điểm chết đơn (no SPOF)
Redundancy	Dư thừa bản sao để chịu lỗi
Failover	Tự chuyển sang bản dự phòng khi lỗi
Replication	Đồng bộ dữ liệu sang bản sao
Disaster Recovery (DR)	Kế hoạch phục hồi sau thảm hoạ
RTO	Recovery Time Objective — bao lâu phục hồi xong
RPO	Recovery Point Objective — chấp nhận mất bao nhiêu data
Backup / Snapshot	Sao lưu định kỳ để khôi phục
Multi-region / Multi-AZ	Trải nhiều vùng để cách ly lỗi

Tool: Cloud auto-scaling groups · K8s HPA/Cluster Autoscaler · Velero · Kopia.

22

Data, Caching & Messaging

Tầng dữ liệu của hệ thống: DB (primary/replica), cache giảm tải, message queue tách rời service và chịu tải đột biến.

1. Database — primary / replica

Sharding = chia dữ liệu nhiều node.
Backup + migration định kỳ.

2. Cache (cache-aside)

Giảm tải DB,
đáp ứng nhanh.

3. Messaging / Event-driven

async, tách rời,
chịu tải đột biến.

Thuật ngữ

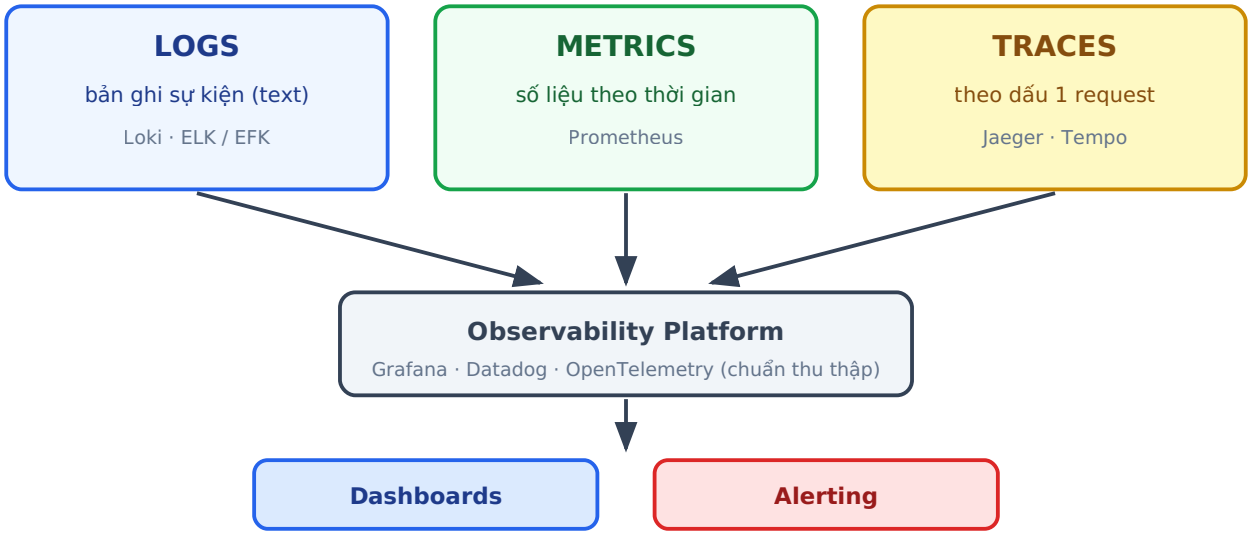
Thuật ngữ	Nghĩa
Primary / Replica	Node ghi chính / bản sao đọc
Replication	Đồng bộ dữ liệu sang replica
Sharding	Chia dữ liệu ra nhiều node
Connection pool	Tái dùng kết nối DB
Migration	Thay đổi schema có version
Cache	Lưu tạm dữ liệu nóng (Redis)
Cache-aside	App đọc cache trước, miss thì xuống DB
TTL / Eviction	Hết hạn / loại bỏ entry cache
Message Queue	Hàng đợi tin nhắn (RabbitMQ)
Pub/Sub	Phát/đăng ký theo topic
Event-driven	Kiến trúc theo sự kiện, async
Stream processing	Xử lý luồng dữ liệu real-time (Kafka)
Backpressure	Cơ chế chống quá tải consumer

Tool: PostgreSQL/MySQL · MongoDB · Redis · Kafka · RabbitMQ · NATS.

23

Observability — 3 trụ cột

Monitoring = theo dõi chỉ số đã biết trước. **Observability** = hiểu trạng thái nội bộ hệ thống từ output, kể cả lỗi chưa lường. Ba trụ cột: Logs, Metrics, Traces.



Thuật ngữ

Thuật ngữ	Nghĩa
Monitoring	Theo dõi chỉ số đã biết trước (CPU, RAM)
Observability	Hiểu trạng thái nội bộ từ output, kể cả lỗi chưa lường
Metrics	Số liệu theo thời gian (req/s, CPU%)
Logs	Bản ghi sự kiện dạng text
Traces	Theo dấu 1 request qua nhiều service
APM	Application Performance Monitoring
Alerting	Cảnh báo khi vượt ngưỡng
Dashboard	Bảng trực quan hoá số liệu
Cardinality	Số giá trị unique của 1 label (cao → tốn tài nguyên)
OpenTelemetry	Chuẩn thu thập telemetry thống nhất

Tool: Prometheus · Grafana · Loki · Jaeger/Tempo · ELK/EFK · Datadog.

24

SRE — SLA / SLO / SLI & DORA

SRE = áp dụng kỹ thuật software vào vận hành. Đo độ tin cậy bằng 3 lớp cam kết và 4 chỉ số DORA.

SLA ≥ SLO ≥ SLI + Error Budget

Error Budget = 100% − SLO

Còn budget → thoải mái ship feature. Hết budget → đóng băng feature, ưu tiên ổn định.

DORA — 4 chỉ số vàng đo năng lực

Thuật ngữ

Thuật ngữ	Nghĩa
SRE	Áp dụng kỹ thuật software vào vận hành
SLA	Cam kết với khách hàng (có phạt nếu vi phạm)
SLO	Mục tiêu nội bộ, chặt hơn SLA
SLI	Chỉ số đo thực tế (% request thành công)
Error budget	Ngân sách lỗi = 100% − SLO
MTTR	Thời gian trung bình phục hồi
MTBF	Thời gian trung bình giữa 2 lần lỗi
Toil	Việc tay lặp lại cần automate
Incident	Sự cố
Postmortem	Báo cáo phân tích sau sự cố (blameless)
On-call	Trực sự cố
Runbook	Quy trình xử lý sự cố từng bước

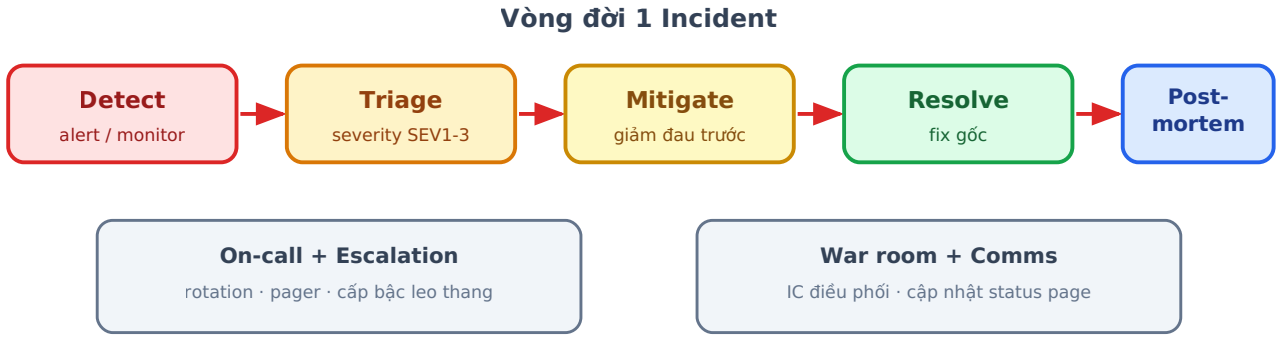
Thuật ngữ	Nghĩa
Chaos Engineering	Cố tình gây lỗi để test độ bền
DORA metrics	4 chỉ số: Deploy Freq · Lead Time · Change Fail Rate · MTTR

Tool: PagerDuty · Opsgenie · Prometheus Alertmanager · Chaos Monkey/LitmusChaos.

25

Incident Management & On-call

Khi sự cố xảy ra: phát hiện → phân loại → giảm đau → fix gốc → mổ xẻ. On-call rotation + escalation đảm bảo luôn có người xử lý.



Blameless postmortem → action items → giảm tái diễn. Runbook = kịch bản xử lý sẵn.

Thuật ngữ

Thuật ngữ	Nghĩa
Incident	Sự cố làm gián đoạn dịch vụ
Severity (SEV1-3)	Mức nghiêm trọng, ưu tiên xử lý
Triage	Phân loại + định mức độ
Mitigation	Giảm đau tạm trước khi fix gốc
On-call	Trực sự cố theo ca
Rotation	Luân phiên ca trực
Escalation	Leo thang lên cấp cao hơn
Incident Commander (IC)	Người điều phối xử lý
War room	Kênh tập trung xử lý sự cố
Status page	Trang công bố tình trạng cho user
Postmortem	Báo cáo sau sự cố (blameless)
Runbook	Kịch bản xử lý sẵn từng bước
Action item	Việc cần làm để chống tái diễn

Tool: PagerDuty · Opsgenie · Incident.io · Statuspage · Slack.

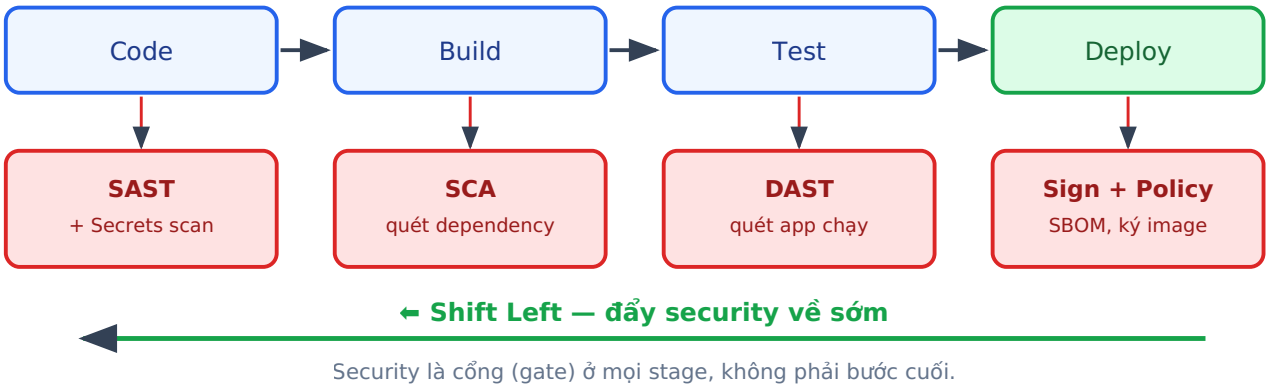
PHẦN

V — Bảo mật · Governance · Thực hành

26

DevSecOps

Nhúng **security vào toàn pipeline** thay vì kiểm tra cuối. Mỗi stage có 1 cổng bảo mật (gate). Triết lý "shift left" — bắt lỗi càng sớm càng rẻ.



Security là cổng (gate) ở mọi stage, không phải bước cuối.

Thuật ngữ

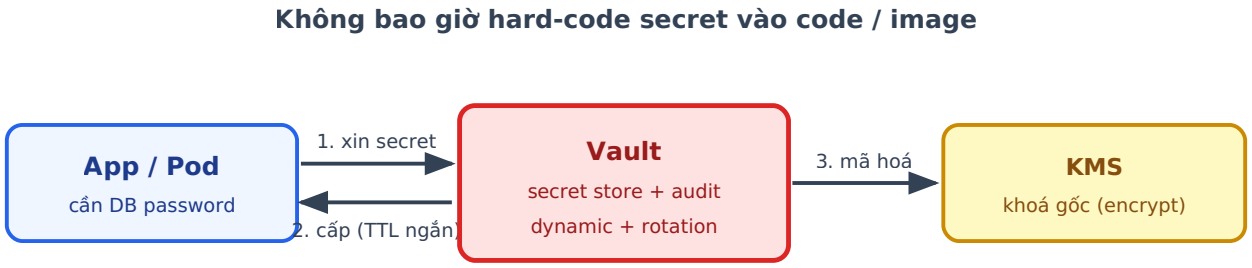
Thuật ngữ	Nghĩa
DevSecOps	Security là trách nhiệm chung, nhúng vào mọi stage
SAST	Quét code tĩnh tìm lỗ hổng (không chạy app)
DAST	Quét app đang chạy từ ngoài vào
SCA	Quét dependency/thư viện có lỗ hổng (CVE)
Secrets management	Quản lý mật khẩu/key an toàn (Vault, SOPS)
SBOM	Software Bill of Materials — kê khai thành phần
Image signing	Ký artifact để chống giả mạo
Least privilege	Cấp quyền tối thiểu đủ dùng
RBAC	Phân quyền theo vai trò
mTLS	Mã hoá + xác thực 2 chiều giữa service
Supply chain security	Bảo vệ chuỗi cung ứng phần mềm
Policy as Code	Quy tắc bảo mật viết bằng code (OPA)

Tool: Trivy · Snyk · SonarQube · HashiCorp Vault · OWASP ZAP · OPA/Gatekeeper.

27

Secrets Management

Mật khẩu/API key/cert **không bao giờ** hard-code vào code hay image. Dùng secret store tập trung (Vault), cấp secret động có thời hạn, xoay khoá định kỳ.



Dynamic secret: sinh credential tạm, tự huỷ sau TTL → giảm rủi ro lộ.
Rotation: xoay khoá định kỳ. Least privilege: chỉ cấp quyền tối thiểu.

Thuật ngữ

Thuật ngữ	Nghĩa
Secret	Dữ liệu nhạy cảm: password, API key, cert
Secret store	Kho lưu tập trung, có audit (Vault)
KMS	Key Management Service — quản lý khoá gốc
Encryption at rest / in transit	Mã hoá lúc lưu / lúc truyền
Dynamic secret	Sinh credential tạm, tự huỷ sau TTL
Rotation	Xoay khoá/mật khẩu định kỳ
Sealed Secrets	Secret mã hoá an toàn để lưu trong Git
Least privilege	Cấp quyền tối thiểu đủ dùng
Audit log	Ghi lại ai truy cập secret nào
Secret scanning	Quét code/lịch sử Git tìm secret lộ

Tool: HashiCorp Vault · AWS/GCP KMS · SOPS · Sealed Secrets · External Secrets Operator.

28

Network Security & Zero Trust

Bảo vệ ở tầng mạng/hạ tầng (khác DevSecOps lo tầng app). Phòng thủ nhiều lớp + nguyên tắc Zero Trust: không tin theo vị trí mạng, verify mọi request.

Lớp phòng thủ mạng (defense in depth)

Zero Trust: "never trust, always verify" — không tin theo vị trí mạng

Kèm: VPN · network segmentation · IDS/IPS · mTLS · security group · least-privilege.

Mọi truy cập đều xác thực + uỷ quyền + mã hoá, dù trong hay ngoài mạng nội bộ.

Thuật ngữ

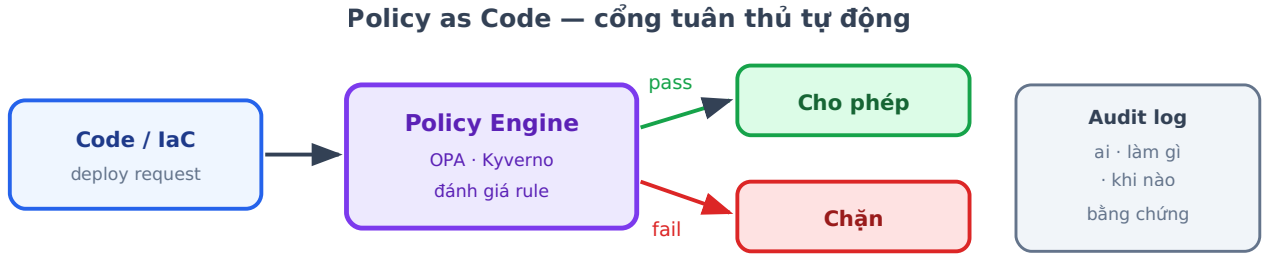
Thuật ngữ	Nghĩa
Firewall	Lọc traffic theo port/IP (L3-4)
WAF	Web App Firewall — lọc tầng app (L7)
DDoS protection	Chống tấn công tràn lưu lượng
Zero Trust	"Never trust, always verify"
VPN	Kênh mã hoá vào mạng riêng
Segmentation	Chia mạng thành vùng cô lập
Security Group	Firewall ảo cấp instance (cloud)
IDS / IPS	Phát hiện / ngăn xâm nhập
mTLS	Mã hoá + xác thực 2 chiều
Bastion / Jump host	Cổng trung gian vào hạ tầng
Least privilege	Quyền tối thiểu đủ dùng
MFA	Xác thực đa yếu tố

Tool: Cloudflare · iptables/nftables · WireGuard · Tailscale · Falco · Suricata.

29

Compliance & Governance

Doanh nghiệp phải tuân thủ chuẩn (SOC 2, GDPR…). DevOps biến tuân thủ thủ công thành **policy as code** — cổng tự động chặn cấu hình vi phạm + ghi audit liên tục.



Governance = rule + audit + bằng chứng tự động hoá; biến tuân thủ thủ công thành code kiểm tra liên tục.

Thuật ngữ

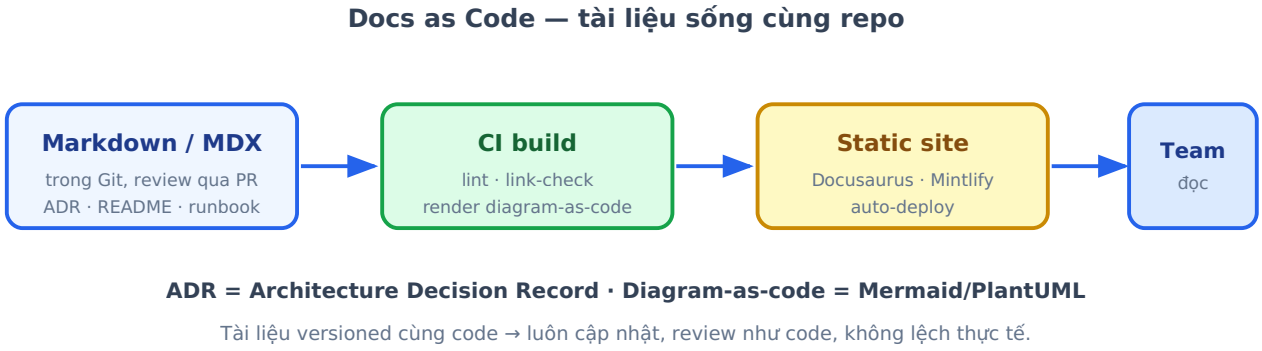
Thuật ngữ	Nghĩa
Compliance	Tuân thủ chuẩn/luật bắt buộc
Governance	Quy tắc + kiểm soát + trách nhiệm
Policy as Code	Quy tắc viết bằng code, kiểm tra tự động
SOC 2	Chuẩn kiểm soát bảo mật dịch vụ
ISO 27001	Chuẩn quản lý an toàn thông tin
GDPR	Luật bảo vệ dữ liệu cá nhân (EU)
HIPAA	Chuẩn dữ liệu y tế (US)
PCI-DSS	Chuẩn dữ liệu thẻ thanh toán
Audit log	Bản ghi ai làm gì khi nào (bằng chứng)
Data residency	Dữ liệu phải nằm ở vùng quy định
Drift / Guardrail	Lệch chuẩn / rào chắn ngăn vi phạm
Evidence collection	Thu thập bằng chứng tuân thủ tự động

Tool: OPA/Gatekeeper · Kyverno · Cloud Custodian · Vanta · Drata.

30

Documentation as Code

Tài liệu viết bằng Markdown, sống trong Git, review qua PR, build + publish tự động qua CI. Luôn cập nhật, không lệch thực tế.



Thuật ngữ

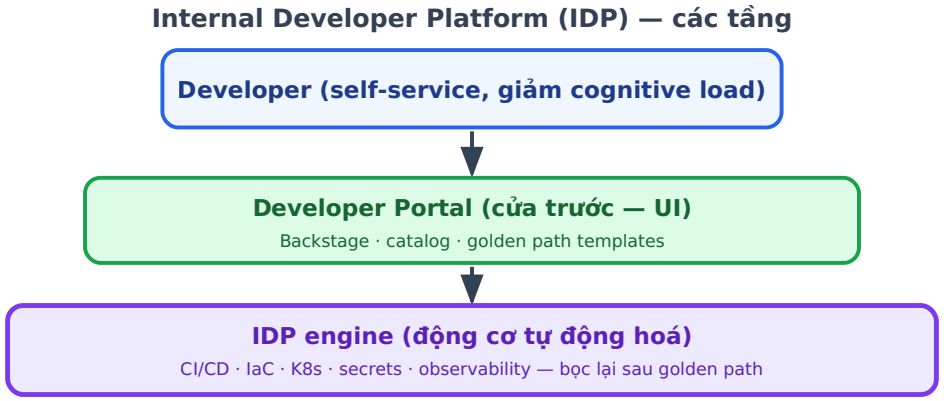
Thuật ngữ	Nghĩa
Docs as Code	Tài liệu trong Git, quy trình như code
Markdown / MDX	Định dạng viết docs
ADR	Architecture Decision Record
README	Tài liệu gốc của repo
Runbook	Hướng dẫn vận hành/xử lý sự cố
Diagram as Code	Vẽ sơ đồ bằng text (Mermaid, PlantUML)
Static site generator	Build docs thành web (Docusaurus)
Single source of truth	1 nguồn tài liệu duy nhất
llms.txt	Index tài liệu thân thiện cho LLM
Changelog	Lịch sử thay đổi có version
Knowledge base	Kho tri thức team

Tool: Docusaurus · Mintlify · MkDocs · Backstage TechDocs · Mermaid.

31

Platform Engineering & IDP

Xu hướng kế thừa DevOps (2025): team platform xây **Internal Developer Platform** gói sẵn CI/CD + infra + secrets thành **golden path** self-service, để dev tập trung viết tính năng.



Golden path = đường mòn đã lát sẵn, không bắt buộc nhưng tốt đến mức ai cũng tự chọn.

Thuật ngữ

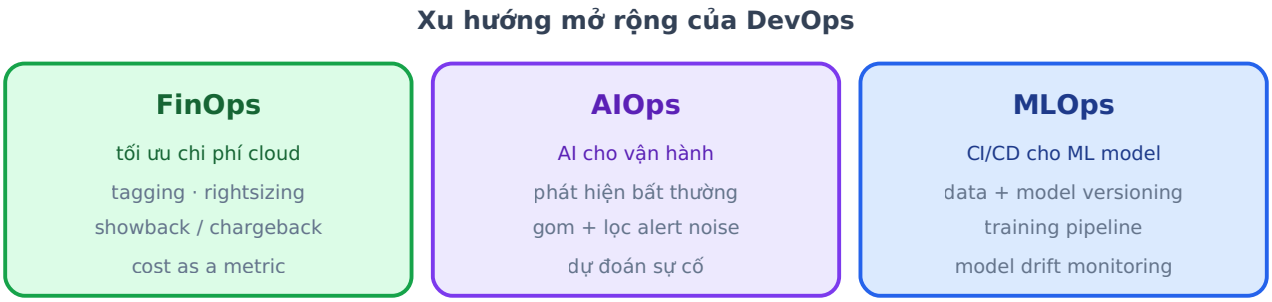
Thuật ngữ	Nghĩa
Platform Engineering	Xây nền tảng nội bộ cho dev tự phục vụ
IDP (Internal Developer Platform)	Động cơ: tự động hoá + infra + golden path
Developer Portal	Cửa trước (UI) để dev tương tác (Backstage)
Golden Path	Đường mòn lát sẵn cho task phổ biến
Self-service	Dev tự cấp tài nguyên không cần ticket
Paved road	Lối đi an toàn, defaults tốt sẵn
Cognitive load	Gánh nặng nhận thức cần giảm cho dev
Service catalog	Danh mục service + ownership
Scaffolding / Template	Khung tạo service mới nhanh
Abstraction	Giấu phức tạp hạ tầng khỏi dev

Tool: Backstage · Port · Humanitec · Crossplane · Cyclops.

32

Xu hướng: FinOps · AIOps · MLOps

DevOps lan sang các nhánh chuyên biệt: tối ưu chi phí (FinOps), AI cho vận hành (AIOps), và CI/CD cho machine learning (MLOps).



Thuật ngữ

Thuật ngữ	Nghĩa
FinOps	Quản lý + tối ưu chi phí cloud như 1 metric
Rightsizing	Chọn đúng kích cỡ tài nguyên, tránh lãng phí
Showback / Chargeback	Báo / tính chi phí về từng team
Tagging	Gắn nhãn tài nguyên để truy chi phí
AIOps	Dùng AI/ML cho vận hành (anomaly, alert)
Anomaly detection	Phát hiện bất thường tự động
Alert correlation	Gom + lọc alert noise
MLOps	CI/CD + vận hành cho ML model
Model / Data versioning	Quản lý version model + dataset
Feature store	Kho feature dùng lại cho ML
Model drift	Model giảm chính xác theo thời gian
GitOps for ML	Khai báo pipeline ML qua Git
GreenOps	Tối ưu năng lượng/carbon của hạ tầng
DataOps	Áp DevOps cho pipeline dữ liệu (CI/CD data)
LLMOps	Vận hành ứng dụng LLM (prompt, eval, cost)
Observability-driven	Dùng telemetry dẫn dắt quyết định vận hành

Tool: OpenCost/Kubecost · CloudHealth · Dynatrace (AIOps) · MLflow · Kubeflow · DVC.

> Đây là phần ngọn đang mở rộng — gốc vẫn là các nguyên lý DevOps ở những chương trước.

Index

N

O

P

R

S

T

U

V

W

Y

Z